Via E-File

PATENT APPLICATION
Docket: 13768.132

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND
INTERFERENCES

In re application of		)
	Damon Barry et al.	)
Serial No.:	09/607,397	) Art Unit
Filed:	June 30, 2000	) 2192
Conf. No.:	9886	)
For:	SYSTEMS AND METHODS FOR GATHERING ORGANIZING AND EXECUTING TEXT CASES	)
Examiner:	Eric B. Kiss	)
Customer No.:	47973	)

BRIEF OF APPELLANTS

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

On September 15, 2006, Appellant timely filed a Notice of Appeal and Pre-appeal Brief Request for Review from the action of the Examiner finally rejecting claims 1-4, 7, 10-17, and 20-28 in this application. A Notice of Panel Decision from Pre-Appeal Brief Review was mailed on November 3, 2006. This appeal brief is being filed under the provisions of 37 C.F.R. § 41.37. The filing fee of $500.00, as set forth in 37 C.F.R. § 41.20(b)(2), is submitted herewith. This brief is being filed on January 3, 2007, accompanied by a one-month Request for Extension of

Time and fee payment, and is therefore timely under 37 C.F.R. § 41.37(a)(1) and 35 U.S.C. § 21(b).

REAL PARTY IN INTEREST

The real party in interest is Microsoft Corporation, by way of assignment from Damon Barry, Greg Veith, Brent Jensen, and Frank Truong, who are the only named inventors. The assignment documents were recorded at Reel No. 010935, Frame 0609 in the United States Patent and Trademark Office on September 12, 2000.

RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

STATUS OF CLAIMS

The application was originally filed with claims 1-27. Independent claims 1, 4, 15, and 24 and dependent claims 2, 3, 12, 14, 21, 23, 26, and 27 were amended and claims 5 and 6 were cancelled by Amendment "A" dated July 2, 2003. Independent claims 1, 4, 15, and 24 were amended by Amendment "B" dated February 4, 2004. Independent claims 1, 4, 15, and 24 and dependent claims 7, 18-21, and 27 were amended by Amendment "C" dated September 16, 2004. Independent claims 1, 4, 15, and 24 were amended by Amendment "D" dated March 31, 2005. Independent claims 1, 4, 15, and 24 and dependent claims 10 were amended, claims 8, 9, 18, and 19 were canceled, and claim 28 was added by Amendment "E" dated July 13, 2005. Independent claim 15 was amended by Amendment "G" dated July 26, 2006. Pending claims 1-4, 7, 10-17, and 20-28 stand rejected and have been appealed.

STATUS OF AMENDMENTS

Amendments "A" through "G" have all been entered by the Examiner, and claims 1-4, 7, 10-17, and 20-28 are presented on appeal in the same form as that finally rejected by the Examiner.

SUMMARY OF INVENTION

As described in the Specification, the present invention relates to a computer system for selecting and organizing individual test cases for use in testing a computer program to ensure that the program processes as intended. *See* 4:8-10. The pending claims include four independent claims, 1, 4, 15, and 24. In one embodiment, as claimed by independent claim 1, the system includes one or more program modules storing one or more available test cases. *Id.* Each test case comprises a set of instructions, known as a harness, for testing a feature of the computer program through a language and format independent interface, and a harness client. 13:3-7; 12:14-17. The harness client comprises a set of instructions that (i) receives user input specifying one or more filenames corresponding to the one or more program modules, (ii) executes a connector to scan for and discover the one or more available test cases that are stored in the one or more program modules and to organize the one or more available test cases into a test case hierarchy, and (iii) receives user input indicating which of the one or more available test cases in the test case hierarchy are selected to be executed on the computer program. *See* 17:1-18:7; 5:7-21. In the test case hierarchy, one or more test cases comprise a test suite, one or more test suites comprises a test module. 4:16-18; *see also* 17:13-18:2. The system also includes a harness comprising a set of instructions that (i) receives the test case hierarchy, (ii) traverses the test case hierarchy, and (iii) executes each of the one or more available test cases that is selected

to be executed on the computer program using the corresponding language and format independent interface of the selected test case to ensure that the computer program processes as intended. *Id.* Also included in the system is a connector comprising a set of instructions that (i) scans for the one or more available test cases stored in the one or more program modules, (ii) organizes the one or more available test cases into the test case hierarchy by extracting the one or more available test cases from the one or more program modules, and (iii) selectively integrates an interface between the test case hierarchy and the harness regardless of the language or format in which the one or more available test cases were written. *See* 15:14-18; *see also* 4:13-18. Additionally, a processor for executing each selected test case, the harness, the harness client, and the connector is included in the system. 10:7-10.

In the computer system, the set of instructions of the harness and the set of instructions of the connector may utilize an architecture that defines a means for accessing a resource over a network. *See* 11:18-24. In such systems, the architecture may be COM technology. *Id.* at 4:7

In embodiments of the invention as claimed in claim 4, a computer system that includes a processor, a computer program to be tested, a program module storing one or more test cases of interest for use in testing the computer program, a harness for executing individual test cases on the computer program, a harness client for receiving user input, and a connector for interfacing between the one or more test cases of interest and the harness, is used to perform a method for testing the computer program to determine whether the computer program processes as intended. 4:2-18. The method includes the harness client (i) receiving user input that specifies one or more filenames to identify the program module, (ii) executes the connector to scan for and discover the one or more test cases of interest that are stored in the program module and to organize the one or more test cases of interest into a test case hierarchy in which one or more test cases comprise a

test suite, and in which one or more test suites comprise a test module, and (iii) receives user input indicating that the one or more test cases of interest in the test case hierarchy are to be executed on the computer program. *See* 17:1-18:7; 5:7-21. The method also includes the connector scanning the one or more test cases of interest stored in the program module. 15:5-10. Each test case has a language and format independent interface for executing the test case on the computer program regardless of the language or format used to develop the test case. *See* 15:14-18; *see also* 4:13-18. In the method, the connector extracts the one or more test cases of interest from the program module, organizes one or more test cases of interest into the test case hierarchy, and interfaces the harness with the one or more test cases of interest. *See* 15:14-19. The interfacing allows the harness to recognize and execute the one or more test cases of interest regardless of the language or format in which the one or more test cases of interest were developed. *Id.* In the method, the harness traverses the test case hierarchy and executes each of the one or more test cases of interest to test the computer program. 16:9-13.

In some embodiments of the method as claimed in claim 4, the method may further include a step for determining whether one or more of the test cases of interest are identified as being deselected. *See* 17:1-7. The deselected test case is not executed on the computer program. In other embodiments, when the user selects a test suite, the one or more test cases that comprise the test suite, excluding any test cases determined to be deselected, are selected. 17:1-22. Additionally, in some embodiments, when the user selects a test module, the one or more test suites that comprise the test module, excluding any test cases determined to be deselected, are selected. *Id.*

In other embodiments of the invention as claimed in claim 4, the step for traversing may further include executing the one or more test cases on a thread pool comprising one or more

threads. 18:7-14. In such cases, the step for traversing may further include copying a selected test case across all of the one or more threads, such that the selected test case is executed across all of the one or more threads. *Id.* Also, the step for traversing may further include executing a selected test case on one of the threads. *Id.*

In embodiments of the invention as claimed in claim 15, the invention relates to a computer program product for implementing a method for testing a computer program in a computer system to determine whether the computer program processes as intended. *See* 4:8-10. In such embodiments, the computer program product includes computer readable physical storage medium for providing computer program code means utilized to implement the method. 8:19-9:10. The computer program code means is comprised of executable code for implementing acts. *Id.*

The acts include receiving user input that specifies one or more filenames to identify one or more program modules storing one or more test cases of interest. 17:1-7. Each test case of interest comprises a set of instructions for testing a feature of the computer program through a language and format independent interface. *See* 15:14-18; *see also* 4:13-18. The acts also include executing a connector to scan for and discover the one or more test cases of interest that are stored in the program module and to organize the one or more test cases of interest into a test case hierarchy. *See* 17:1-18:7. In the hierarchy, one or more test cases comprise a test suite, and one or more test suites comprise a test module. 5:7-21. Other acts include receiving user input indicating that the one or more test cases of interest in the test case hierarchy are to be executed on the computer program, scanning for the one or more test cases of interest that are stored in the one or more program modules, extracting the one or more test cases of interest from the program module, organizing the one or more test cases of interest into the test case hierarchy, interfacing

with the one or more test cases of interest to recognize and execute the one or more test cases of interest regardless of the language or format in which the one or more test cases of interest were developed, traversing the test case hierarchy, and executing the one or more test cases of interest on the computer program. 17:1-18:7.

In such embodiments the computer program product, the step for traversing may be performed by the harness. 12:17-21. Additionally, the step for interfacing may be performed by one or more connectors. *See* 4:8-14. In some embodiments, the one or more test cases executed on the computer program are selected by a user through a user interface provided by a harness client. 13:3-7. The step for traversing may further include executing the one or more selected test cases on a thread pool comprising one or more threads, and copying a selected test case across all of the one or more threads, such that the selected test case is executed across all of the one or more threads. 18:7-14. Additionally, the step for traversing may further include executing a selected test case on one of the threads. *Id.*

In some embodiments, as claimed in claim 24, a computer system that includes a computer program to be tested, a program module of one or more test cases written in any format or language for testing the computer program, a harness for executing one test cases on the computer program, a harness client for receiving user input, and one or more connectors for interfacing test cases with the harness performs a method for testing the computer program to determine whether the computer program processes as intended. *See* 4:2-18. Such a method includes specifying one or more filenames for identifying one or more program modules storing one or more test cases, each test case comprising a set of instructions for testing a feature of the computer program through a language and format independent interface. 15:18-19; *see also* 16:1-8. Other steps include identifying the one or more test cases within the one or more

program modules, translating the identified one or more test cases into a test case hierarchy. *See* 17:1-22. In the test case hierarchy, one or more test cases comprise a test suite, and one or more test suites comprise a test module. 4:16-18; *see also* 17:13-18:2. Other steps included are indicating that the one or more test cases in the test case hierarchy are to be executed on the computer program, providing an interface to the test case hierarchy in order to recognize and execute the one or more test cases regardless of the language or format in which the one or more test cases were written, and running each of the one or more test cases in the test case hierarchy to test the computer program. *Id.*

In such embodiments, the act of executing may be performed on a thread pool comprising one or more threads, and the act of executing may be further performed by executing a selected test case on one of the threads. 18:7-14. In other such embodiments, the act of executing is further performed by copying the one or more selected test cases across all of the one or more threads, such that the selected test case is executed across all of the one or more threads. *Id.*

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

In the Final Office Action mailed August 15, 2006 ("Final Office Action")[1], claims 1, 2, 4, 7, 10-17, and 20-28 were rejected under 35 U.S.C. 102(b) as being anticipated by the TETware Release 3.3 software product released September 18, 1998 by The Open Group, as evidenced by: "TETware User Guide, Revision 1.2", "Release Notes for TETware Release 3.3" and "TETware Programmers Guide, Revision 1.2"; and claim 3 was rejected under 35 U.S.C. 103(a) as being unpatentable over TETware and the associated cited documentation as applied to claim 1 above, and further in view of Hartman et al. (U.S. Patent No. 6,505,342).

[1] Unless specifically referenced, "Office Action" or "Final Office Action" refers to the Final Office Action mailed August 15, 2006.

ARGUMENT

A. Claims 1, 2, 4, 7, 10-17, and 20-28 are not anticipated by TETware Release 3.3 because the asserted prior art does not disclose each and every element of independent claims 1, 4, 15, and 24.

TETware fails to disclose or suggest each and every element of the claimed invention. "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." MPEP § 2131. That is, "for anticipation under 35 U.S.C. 102, the reference must teach every aspect of the claimed invention either explicitly or impliedly." MPEP § 706.02.

1. TETware does not disclose the claimed test hierarchy

TETware does not disclose a "test hierarchy in which one or more test cases comprise a test suite, and in which one or more test suites comprise a test module," as recited in each of independent claims 1, 4, 15, and 24. As is clear from the plain language of the claims, a test hierarchy having at least three distinct levels is claimed. A **test module**, which contains at least one **test suite**, which contains at least one **test case**, is claimed.

The Examiner alleges that "TETware [discloses] one or more test cases comprising a test suite in the hierarchy and one or more test suites comprising a test module in the hierarchy," citing section 2.2 of TETware User Guide, 4.1 of TETware Programmers Guide, and 5.3.2.1 of TETware User Guide. Final Office Action at 4. A careful review of the TETware literature reveals that the cited sections simply ***do not*** disclose or suggest a "test hierarchy in which ***one or more test cases comprise a test suite***, and in which ***one or more test suites comprise a test module***," as recited in independent claims 1, 4, 15, and 24. (Emphasis added). Instead, TETware contradicts the Examiner, clearly stating that TETware has exactly two levels in a test hierarchy, and not three as claimed.

The relevant portions of the portions of the TETware documentation cited in the Final Office Action are recited below to highlight the Examiner's error in interpreting the reference.

> "A **test suite is the largest grouping of tests** that can be processed by the TETware Test Case Controller. **A test suite is made up of one or more test cases.** A **test case is the smallest test program unit** that can be built or cleaned up by the Test Case Controller."

TETware User Guide § 2.2 (emphasis added).

> "The scenario file contains one or more test scenarios for a test suite… A scenario file should contain (at least) a scenario named "all"; by convention, this causes **all the test cases in the test suite** to be processed."

TETware User Guide § 5.3.2.1 (emphasis added).

> "When the tcc processes test cases, it does so by reading instructions contained in a test scenario. Each test suite should include a scenario file which contains one or more test scenarios."

TETware Programmers Guide § 4.2.

TETware calls its highest level a "test suite," and its lowest level a "test case," with "test suites" having "test cases." There is simply no discussion of a third level corresponding to a "test module" in TETware. Instead, TETware discusses only a separate instruction hierarchy including scenario files and test scenarios.

A TETware "test scenario" shares no hierarchical position with "test suite" and "test case," but is rather separate instructions for using "test suite" and "test case." Section 2.2 of the User Guide makes clear the hierarchical organization of TETware, fully contradicting the Examiner's position that TETware discloses the claimed elements. "A **test suite is the largest grouping of tests** that can be processed by the TETware Test Case Controller. **A test suite is**

made up of one or more test cases. A **test case is the smallest test program unit** that can be built or cleaned up by the Test Case Controller." TETware User Guide § 2.2 (emphasis added).

The scenario is merely a set of instructions within the test suite to determine which of the test cases in the test suite to run. Instructions are not "tests" and do not occupy any hierarchical position between, above, or below the test case or the test suite. Instructions simply aid in determining which test cases in the test suite are applied to a particular situation, and, thus, do not occupy any hierarchical position in TETware.

As discussed, TETware simply does not disclose or suggest a "test hierarchy in which one or more test cases comprise a test suite, and in which one or more test suites comprise a test module," as recited in each of independent claims 1, 4, 15, and 24. As such, TETware does not anticipate each and every element of the claimed invention under 35 U.S.C. § 102(b).

2. TETware does not disclose the claimed test case discovery

TETware does not disclose "a harness client comprising a set of instructions that (i) receives user input specifying one or more filenames corresponding to the one or more program modules, (ii) executes a connector to scan for and discover the one or more available test cases that are stored in the one or more program modules and to organize the one or more available test cases into a test case hierarchy in which one or more test cases comprise a test suite, and in which one or more test suites comprise a test module, and (iii) receives user input indicating which of the one or more available test cases in the test case hierarchy are selected to be executed on the computer program," as recited in claim 1. Similar recitations are found in independent claims 4, 15, and 24. For claims 1, 4 and 15, the claimed instructions relate to receiving user input that specifies one or more filenames corresponding to one or more program modules, executing a connector to scan for and discover one or more available test cases that are

stored in the one or more program modules and to organize the one or more available test cases into a test case hierarchy, and receiving user input that indicates which of the one or more available test cases in the test case hierarchy are selected to be executed on the computer program being tested. For independent claim 24, the instructions relate to specifying one or more filenames for identifying one or more one or more program modules storing one or more test cases, each comprising a set of instructions for testing a feature of the computer program through a language and format independent interface, and identifying the one or more test cases within the one or more program modules.

The Examiner alleges that the claimed features are described in section 5.3.2 of TETware UG, without referring to specific language in that section to support that assertion. *See, e.g.,* Office Action, October 6, 2005, pp. 3-4. In spite of repeated requests for clarification (*See, e.g.,* Amendment "E", pg. 13, Amendment "F", pg. 14), the Examiner simply repeated the same allegation of anticipation. *See, e.g.,* Office Action, October 6, 2005, pg. 1.

After reviewing section 5.3.2, it appears that the section just describes the scenario file, which seems to include scenario directives. There is simply no discussion in TETware TETware User Guide § 5.3.2 describing how the scenario directives cause the recited test case instruction functionality now recited in the claims. The Examiner seems to imply that the disclosed scenario file name in section 5.3.2 corresponds to the claim requirement of one or more filenames corresponding to one or more program modules. Office Action, October 6, 2005, pp. 3-4. However, nowhere in cited TETware documentation does it teach or disclose that the scenario file name is used to identify a program module that stores one or more test cases. TETware User Guide § 5.3.2 and TETware Programmers Guide § 4.1 discuss a scenario file name used to indicate which scenario to process, but does not discuss identifying a program module.

In fact, if the program module were the test suite as indicated by the Examiner, the cited portions of TETware teach away from using the scenario name to identify the test suite because the test suite already includes at least a scenario name "all." Consequently there would be no need to identify the test suite by the recited filename.

Thus, TETware simply does not disclose the test instructions as claimed in each of independent claims 1, 4, 15, and 24. As such, TETware does not anticipate each and every element of the claimed invention under 35 U.S.C. § 102(b).

Each of claims 2, 7, 10-14, 16, 17, 20-23, and 25-28 depends from one of independent claims 1, 4, 15, and 24 either directly or through another claim. Each dependent claim includes the limitations of the claim or claims from which they depend. Thus, each of the dependent claims is allowable at least for the same reasons as independent claims 1, 4, 15, and 24. Consequently, the § 102(b) rejection of claims 1, 2, 4, 7, 10-17, and 20-28 should be withdrawn and the claims allowed as presented.

B. No *prima facie* case of obviousness under 35 U.S.C. § 103(a) exits in the rejection of claim 3 because the asserted prior art does not disclose or suggest each and every element of independent claims 1, 4, 15, and 24.

"To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations." M.P.E.P. § 2143.

No *prima facie* case of obviousness was established at least because there is no disclosure or suggestion of each and every claim limitation in the cited references. In rejecting claim 3 under 35 U.S.C. § 103(a) as being unpatentable over TETware in view of U.S. Patent No.

6,505,342 to Hartman et al. ("Hartman"), the Examiner applied the reasoning of the rejection of claim 1 under 35 U.S.C. §102(b), as discussed above, to the rejection under 35 U.S.C. § 103(a). *See* Office Action at 10. Thus, TETware fails to disclose each element of the claimed invention as discussed above. Also, Hartman does not disclose a "test hierarchy in which one or more test cases comprise a test suite, and in which one or more test suites comprise a test module," and is not cited by the Examiner as doing so. There also no suggestion in the prior art of a "test hierarchy in which one or more test cases comprise a test suite, and in which one or more test suites comprise a test module," as claimed in claim 1.

As discussed above, TETware expressly indicates that only two hierarchical test levels exist. This express indication of only two levels precludes any suggestion of more than two levels. As such, neither TETware nor Hartman, either alone or in combination, discloses or suggests each and every element of the claimed invention. Consequently, the Examiner has failed to establish a *prima facie* case of obviousness as to claim 3. Accordingly, the Examiner's rejection of claim 3 should be removed and those claims should be allowed as presently written.

CONCLUSION

For the foregoing reasons, Appellant respectfully requests the Board to overturn the Examiner's rejections of the appealed claims 1-4, 7, 10-17, and 20-28.

Dated this 3rd day of January, 2007.

Respectfully submitted,

WORKMAN NYDEGGER

/F. Chad Copier/ Reg. No. 54,047
RICK D. NYDEGGER
Registration No. 28,651
ADRIAN J. LEE
Registration No. 42,785
F. CHAD COPIER
Registration No. 54,047
Customer No. 047973
Attorneys for Applicant

FCC:jbh
FCC0000000036V001

CLAIMS APPENDIX

1. A computer system for selecting and organizing individual test cases for use in testing a computer program to ensure that the program processes as intended, the system comprising:

one or more program modules storing one or more available test cases, each comprising a set of instructions for testing a feature of the computer program through a language and format independent interface;

a harness client comprising a set of instructions that (i) receives user input specifying one or more filenames corresponding to the one or more program modules, (ii) executes a connector to scan for and discover the one or more available test cases that are stored in the one or more program modules and to organize the one or more available test cases into a test case hierarchy in which one or more test cases comprise a test suite, and in which one or more test suites comprise a test module, and (iii) receives user input indicating which of the one or more available test cases in the test case hierarchy are selected to be executed on the computer program;

a harness comprising a set of instructions that (i) receives the test case hierarchy, (ii) traverses the test case hierarchy, and (iii) executes each of the one or more available test cases that is selected to be executed on the computer program using the corresponding language and format independent interface of the selected test case to ensure that the computer program processes as intended;

a connector comprising a set of instructions that (i) scans for the one or more available test cases stored in the one or more program modules, (ii) organizes the one or more available test cases into the test case hierarchy by extracting the one or more available test cases from the one or more program modules, and (iii) selectively integrates an interface between the test case hierarchy and the harness regardless of the language or format in which the one or more available test cases were written; and

a processor for executing each selected test case, the harness, the harness client, and the connector.

2. A computer system as recited in claim 1, wherein the set of instructions of the harness and the set of instructions of the connector utilize an architecture that defines a means for accessing a resource over a network.

3. A computer system as recited in claim 2, wherein the architecture is COM technology.

4. In a computer system that includes a processor, a computer program to be tested, a program module storing one or more test cases of interest for use in testing the computer program, a harness for executing individual test cases on the computer program, a harness client for receiving user input, and a connector for interfacing between the one or more test cases of interest and the harness, a method for testing the computer program to determine whether the computer program processes as intended, the method comprising the acts of:

the harness client (i) receiving user input that specifies one or more filenames to identify the program module, (ii) executes the connector to scan for and discover the one or more test cases of interest that are stored in the program module and to organize the one or more test cases of interest into a test case hierarchy in which one or more test cases comprise a test suite, and in which one or more test suites comprise a test module, and (iii) receives user input indicating that the one or more test cases of interest in the test case hierarchy are to be executed on the computer program;

the connector scanning the one or more test cases of interest stored in the program module, each test case having a language and format independent interface for executing the test case on the computer program regardless of the language or format used to develop the test case;

the connector extracting the one or more test cases of interest from the program module;

the connector organizing one or more test cases of interest into the test case hierarchy;

the connector interfacing the harness with the one or more test cases of interest, wherein the interfacing allows the harness to recognize and execute the one or more test cases of interest regardless of the language or format in which the one or more test cases of interest were developed; and

the harness traversing the test case hierarchy and executing each of the one or more test cases of interest to test the computer program.

7. A method as recited in claim 4, wherein the method further includes the step for determining whether one or more of the test cases of interest are identified as being deselected, wherein a deselected test case is not executed on the computer program.

10. A method as recited in claim 7, wherein upon the user selecting a test suite, the one or more test cases that comprise the test suite, excluding any test cases determined to be deselected are selected.

11. A method as recited in claim 10, wherein upon the user selecting a test module, the one or more test suites that comprise the test module, excluding any test cases determined to be deselected are selected.

12. A method as recited in claim 4, wherein the step for traversing further includes executing the one or more test cases on a thread pool comprising one or more threads.

13. A method as recited in claim 12, wherein the step for traversing further includes copying a selected test case across all of the one or more threads, and wherein the selected test case is executed across all of the one or more threads.

14. A method as recited in claim 12, wherein the step for traversing further includes executing a selected test case on one of the threads.

15. A computer program product for implementing within a computer system a method for testing a computer program to determine whether the computer program processes as intended, the computer program product comprising:

computer readable physical storage medium for providing computer program code means utilized to implement the method, wherein the computer program code means is comprised of executable code for implementing the acts of:

receiving user input that specifies one or more filenames to identify one or more program modules storing one or more test cases of interest, each comprising a set of instructions for testing a feature of the computer program through a language and format independent interface;

executing a connector to scan for and discover the one or more test cases of interest that are stored in the program module and to organize the one or more test cases of interest into a test case hierarchy in which one or more test cases comprise a test suite, and in which one or more test suites comprise a test module;

receiving user input indicating that the one or more test cases of interest in the test case hierarchy are to be executed on the computer program;

scanning for the one or more test cases of interest that are stored in the one or more program modules;

extracting the one or more test cases of interest from the program module;

organizing the one or more test cases of interest into the test case hierarchy;

interfacing with the one or more test cases of interest to recognize and execute the one or more test cases of interest regardless of the language or format in which the one or more test cases of interest were developed;

traversing the test case hierarchy; and

executing the one or more test cases of interest on the computer program.

16. A computer program product as recited in claim 15, wherein the step for traversing is performed by the harness.

17. A computer program product as recited in claim 15, wherein the step for interfacing is performed by one or more connectors.

20. A computer program product as recited in claim 15, wherein the one or more test cases executed on the computer program are selected by a user through a user interface provided by a harness client.

21. A computer program product as recited in claim 20, wherein the step for traversing further includes executing the one or more selected test cases on a thread pool comprising one or more threads.

22. A computer program product as recited in claim 21, wherein the step for traversing further includes copying a selected test case across all of the one or more threads, and wherein the selected test case is executed across all of the one or more threads.

23. A computer program product as recited in claim 21, wherein the step for traversing further includes executing a selected test case on one of the threads.

24. In a computer system that includes a computer program to be tested, a program module of one or more test cases written in any format or language for testing the computer program, a harness for executing one test cases on the computer program, a harness client for receiving user input, and one or more connectors for interfacing test cases with the harness, a method for testing the computer program to determine whether the computer program processes as intended, the method comprising steps for:

specifying one or more filenames for identifying one or more program modules storing one or more test cases, each comprising a set of instructions for testing a feature of the computer program through a language and format independent interface;

identifying the one or more test cases within the one or more program modules;

translating the identified one or more test cases into a test case hierarchy in which one or more test cases comprise a test suite, and in which one or more test suites comprise a test module;

indicating that the one or more test cases in the test case hierarchy are to be executed on the computer program;

providing an interface to the test case hierarchy in order to recognize and execute the one or more test cases regardless of the language or format in which the one or more test cases were written; and

running each of the one or more test cases in the test case hierarchy to test the computer program.

25. A method as recited in claim 24, wherein the act of executing is performed on a thread pool comprising one or more threads.

26. A method as recited in claim 25, wherein the act of executing is further performed by executing a selected test case on one of the threads.

27. A method as recited in claim 25, wherein the act of executing is further performed by copying the one or more selected test cases across all of the one or more threads, and wherein the selected test case is executed across all of the one or more threads.

28. A computing system as recited in claim 1, wherein the test module comprises a plurality of test suites.